zsagent总体框架如图所示:

总共有两个进程在工作:policy\_proc和probe\_service,里面分别有多个线程在相互协作.

policy\_proc中工作机理概要:

两个线程在相互协作，称为thread1和thread2

thread1主要负责和服务器进行通信工作，主要有服务注册，心跳保活，控制策略下载等工作，当控制策略变更时通知thread2;

thread2主要是接收到thread1的通知时进行控制策略解析，并通过IPC发送给probe\_service进程

probe\_service中工作机理概要:

三个线程在相互协作,称为thread1,thread2,thread3

thread1主要负责接受policy\_proc中的策略数据，转化为数据结构存储，加速内核请求处理

thread2主要接收内核的请求数据，根据控制策略来选择放行还是拒绝

thead3主要处理在应用卸载时能够及时更新策略

如何管理控制策略:

1.可信应用运行:应用

和内核数据接口:

主要是根据bfx\_file\_logic接口进行分割的，需要结合接口的数据进行适当修正。

|  |
| --- |
| bfx\_file\_logic(struct task\_struct \*task ,struct file \*file ,int mode ,ulong point) |

请求数据结构，包含请求数据和结果:

|  |
| --- |
| typedef struct bfx\_proc\_s{  pid\_t pid;  pid\_t ppid;  uid\_t uid;  char time[32];  char taskname[256];  char taskpath[BFX\_DEF\_PATH\_LEN];  }bfx\_proc\_t;  typedef struct bfx\_file\_s{  uint isdir;  int mode;  ulong fsize;  ulong rsize;  ulong wsize;  char ftype[16];  char filepath[BFX\_DEF\_PATH\_LEN];  }bfx\_file\_t;  typedef struct bfx\_file\_access\_s{  uint level;  bfx\_proc\_t proc;  bfx\_file\_t file;  }bfx\_file\_access\_t;  typedef struct bfx\_access\_result {  int result;  }bfx\_access\_result\_t;  typedef struct bfx\_2user\_file\_access {  int event;  int subevent;  bfx\_file\_access\_t request;  bfx\_access\_result\_t result;  }bfx\_2user\_file\_access\_t; |

内核需要提供两个接口,其中接口的数据如上所示:

1. 如何获取内核请求

Void \*get\_request();

1. 返回给内核结果

Int send\_response(void \*result);